(Almost) Featureless Stereo –

Calibration and Dense 3D Reconstruction Using Whole Image Operations

V.N. Smelyanskiy

R.D. Morris

D.A. Maluf

P. Cheeseman

RIACS Technical Report 01.26
October 2001

(Almost) Featureless Stereo – Calibration and Dense 3D Reconstruction Using Whole Image Operations[1]

V.N. Smelyanskiy, NASA Ames
R.D. Morris, D.A. Maluf, P. Cheeseman, RIACS

RIACS Technical Report 01.26
October 2001

The conventional approach to shape from stereo is via feature extraction and correspondences. This results in estimates of the camera parameters and a typically sparse estimate of the surface.

Given a set of calibrated images, a *dense* surface reconstruction is possible by minimizing the error between the observed image and the image rendered from the estimated surface with respect to the surface model parameters.

Given an uncalibrated image and an estimated surface, the camera parameters can be estimated by minimizing the error between the observed and rendered images as a function of the camera parameters.

We use a *very small* set of matched features to provide camera parameter estimates for the *initial* dense surface estimate. We then re-estimate the camera parameters as described above, and then re-estimate the surface. This process is iterated. Whilst it can not be proven to converge, we have found that around three iterations results in excellent surface and camera parameter estimates.

[1]This work was supported in part by the National Aeronautics and Space Administration (NASA) under Cooperative Agreement MCC 2-1006 with the Universities Space Research Association (USRA).

(Almost) Featureless Stereo – Calibration and Dense 3D Reconstruction Using Whole Image Operations

V.N. Smelyanskiy
NASA Ames Research Center
Moffett Field, CA, 94035

R.D. Morris, D.A. Maluf and P. Cheeseman
Research Institute for Advanced Computer Science
NASA Ames Research Center
Moffett Field, CA, 94035

October 31, 2001

1 Abstract

The conventional approach to shape from stereo is via feature extraction and correspondences. This results in estimates of the camera parameters and a typically sparse estimate of the surface.

Given a set of calibrated images, a *dense* surface reconstruction is possible by minimizing the error between the observed image and the image rendered from the estimated surface with respect to the surface model parameters.

Given an uncalibrated image and an estimated surface, the camera parameters can be estimated by minimizing the error between the observed and rendered images as a function of the camera parameters.

We use a *very small* set of matched features to provide camera parameter estimates for the *initial* dense surface estimate. We then re-estimate the camera parameters as described above, and then re-estimate the surface. This process is iterated. Whilst it can not be proven to converge, we have found that around three iterations results in excellent surface and camera parameter estimates.

2 Introduction

The goal of surface recovery is to take a set of images and estimate the positions and orientations of the cameras that produced the images, and a representation of the sur-

face that was imaged. This is an example of an *inverse problem*. The forward (or direct) problem is: given a surface and the position and orientation of a camera, what is the expected image? This is the area of computer graphics known as rendering [1]. The inverse problem is: given an set of images, estimate the position and orientation of the cameras, and the shape and reflectance properties of the surface. That is, estimate a *generative model* [2].

We apply Bayesian inference to this problem, as it has been shown to be the natural approach to inverse problems of this kind [3]. We postulate *models* for the surface and for the imaging process, and Bayes theorem tell us how to estimate the parameters of these models from the image data. We use a simple triangulated mesh model for the geometry of the surface, storing heights, z, at each vertex of the mesh. We also associate a parameterized reflectance model with the surface. For simplicity here we consider the Lambertian model, and store a single albedo value, ρ at each vertex. (For multispectral data we store an array of albedo values, one for each spectral band.)

We use the standard pinhole camera model for the image formation process [4], and assume that the internal camera parameters are known. (See, for example, [5] for a simple method of internal camera calibration.) The theoretical development of our approach can be generalized to other imaging geometries and surface reflectance models.

Thus we wish to infer the heights, $\mathbf{z}$, the albedos, ρ and the camera parameters, Θ, from the images. Bayes theorem gives

$$p(\mathbf{z}, \rho, \Theta|\{I\}) \propto p(\{I\}|\mathbf{z}, \rho, \Theta)p(\mathbf{z}, \rho, \Theta) \tag{1}$$

We assume that the priors are independent, so that

$$p(\mathbf{z}, \rho, \Theta) = p(\mathbf{z})p(\rho)p(\Theta)$$

and use a simple smoothness prior for $\mathbf{z}$ and ρ based on penalising curvature, and a uniform prior on Θ. The likelihood is assumed to result from Gaussian errors between the image $\hat{I}(\mathbf{z}, \rho, \Theta)$ synthesized from the surface model and the observed images $\{I\}$, giving

$$p(\{I\}|\mathbf{z}, \rho, \Theta) \propto \exp\left(\sum_{f,p}\left(I_{f,p} - \hat{I}_{f,p}(\mathbf{z}, \rho, \Theta_f)\right)^2/(2\sigma_e^2)\right) \tag{2}$$

where the sum is over all pixels, p in all images, I_f. The surface parameters, $\mathbf{z}$, ρ, are clearly shared between all images. Each image has its own set of camera parameters, Θ_f.

The function $\hat{I}(\mathbf{z}, \rho, \Theta)$ is the process of *rendering* the surface described by $\{\mathbf{z}, \rho\}$ with the camera location and orientation given by Θ. This is clearly nonlinear, and makes optimization of the posterior distribution in equation 1 difficult. To make progress in finding the maximum a-posteriori (MAP) estimate, we linearize the image formation process about the current estimate,

$$\hat{I}(\mathbf{z}, \rho, \Theta) = \hat{I}(\mathbf{u}_0) + \mathbf{D}\mathbf{x} \tag{3}$$

where $\mathbf{u} = \{\mathbf{z}, \rho, \Theta\}$, $\mathbf{x} = \mathbf{u} - \mathbf{u}_0$ and

$$\mathbf{D} = \left\{ \frac{\partial \hat{I}}{\partial \mathbf{z}}, \frac{\partial \hat{I}}{\partial \rho}, \frac{\partial \hat{I}}{\partial \Theta} \right\}$$

If we use a Gaussian smoothness prior with covariance matrix Σ as described above then the linearization converts finding the MAP estimate to the minimization of a quadratic form

$$L = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}\mathbf{x} \tag{4}$$

$$\mathbf{A} = \Sigma^{-1} + \frac{1}{\sigma_e^2}\mathbf{D}\mathbf{D}^T \tag{5}$$

$$\mathbf{b} = \frac{I - \hat{I}(\mathbf{z}, \rho, \Theta)}{\sigma_e^2}\mathbf{D} \tag{6}$$

which is equivalent to the solution of the system of equations

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{7}$$

Consider the structure of this system of equations. The matrix of derivatives $\mathbf{D}$ is of dimensions

$$\begin{aligned}&\text{(no. of pixels)} \times \text{(no. of heights + no. of albedos +} \\ &\qquad\qquad \text{no. of camera parameters)}\end{aligned} \tag{8}$$

or, for the results presented later

$$(256 \times 256) \times (301 \times 301 + 301 \times 301 + 6)$$

The portion of this matrix that is due to the differentials with respect to $\mathbf{z}$ and ρ is very sparse, as typically each mesh vertex is used by a few of the triangles that make up the surface, and these triangles project into only a few pixels. The portion due to the differentials with respect to the camera parameters is, however, dense, as changing any one of the camera parameters typically affects the intensities of all the pixels in the image. As a result of this, $\mathbf{D}\mathbf{D}^T$ and hence $\mathbf{A}$ are very large (around $(180,000 \times 180,000)$ and *dense* (around 3×10^{10} elements). It is clearly impractical to perform joint estimation in this manner. Instead we estimate alternately the camera parameters and the surface parameters, that is

$$\begin{aligned}&\text{given } \Theta, \text{ estimate } \{\mathbf{z}, \rho\} \\ &\text{given } \{\mathbf{z}, \rho\}, \text{ estimate } \Theta\end{aligned} \tag{9}$$

In this way we compute either with a very large, but very sparse matrix when estimating $\mathbf{z}$ and ρ, or with a very small, dense matrix when estimating Θ. The estimates are made by using conjugate gradient to solve equation 7 in an iterative manner. At convergence, we update the current estimate, $\mathbf{u}_1 = \mathbf{u}_0 + \mathbf{x}$, re-render to compute new values of $\hat{I}(\mathbf{z}, \rho, \Theta)$ and $\mathbf{D}$, and repeat the solution of equation 7 until a stable solution is reached.

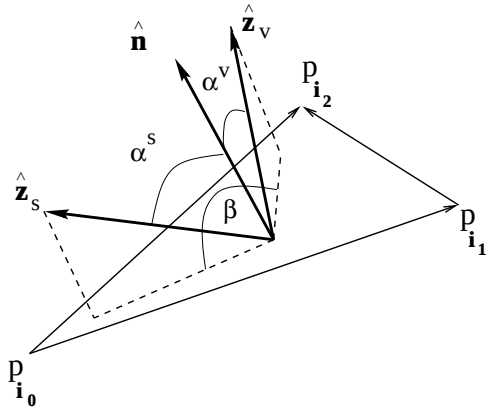

Figure 1: Geometry of the triangular facet, illumination direction and viewing direction. $\hat{\mathbf{z}}_s$ is the vector to the illumination source; $\hat{\mathbf{z}}_v$ is the viewing direction.

This requires an initialization for either Θ or $\{\mathbf{z}, \rho\}$. We use initial values for Θ from point matching a *very small* number of points, or from nominal camera position and orientations, if they are known (eg from rover or aircraft dead-reckoning). In the experiments described later, point matching was used.

3 Forming the Image

As we have seen, to solve the inverse problem we must be able to simulate the forward problem, to compute $\hat{I}(\mathbf{z}, \rho, \Theta)$, ("rendering"). Current rendering technology uses "image space" computation, where the fundamental unit is the pixel. Each pixel is assumed to be illuminated by light from one, and only one, triangular facet. This assumption makes for very fast rendering, but results in aliasing artefacts. It also makes the rendering process non-differentiable.

To enable a renderer to also compute derivatives it is necessary that all computations are done in "object space". This implies that the light from a surface triangle, as it is projected into a pixel, contributes to the brightness of that pixel with a weight proportional to the fraction of the area of the triangle which projects into that pixel. The total brightness of the pixel is thus the sum of the contributions from all the triangles who's projections overlaps with the pixel

$$\hat{I}_p = \sum_{\triangle} f_{\triangle}^p \, \Phi_{\triangle}, \tag{10}$$

where $f_{\triangle}^p$ is the fraction of the flux from triangle $\triangle$ that falls into pixel p, given by

$$f_{\triangle}^p = \frac{\bar{A}_{\text{polygon}}}{\bar{A}_{\triangle}}, \tag{11}$$

where $\bar{A}$ denotes projected area, and $\Phi_{\triangle}$ is the total flux from the triangle, and $\bar{a}_{\text{polygon}}$ is the area on the image plane of the intesection of the projection of the triangle and the pixel. In the case of Lambertian reflection, this is given by

$$\begin{aligned} &\Phi_{\triangle} = \rho\, E(\alpha^s)\, \cos\alpha^v\, (\cos\theta)^{\kappa}\, \Delta\Omega, \\ &E(\alpha^s) = \mathcal{A}\, (\mathcal{I}^s\, \cos\alpha^s + \mathcal{I}^a)\,. \\ &\Delta\Omega = S/d^2. \end{aligned} \tag{12}$$

Here ρ is an average albedo of the triangular facet. Orientation angles α^s and α^v are defined in figure 1. $E(\alpha^s)$ is the total radiation flux incident on the triangular facet with area $\mathcal{A}$. This flux is modeled as a sum of two terms. The first term corresponds to direct radiation with intensity $\mathcal{I}^s$ from the light source at infinity (commonly the sun). The second term corresponds to ambient light with intensity $\mathcal{I}^a$. The parameter θ in equation (12) is the angle between the camera axis and the viewing direction (the vector from the surface to the camera); κ is the lens falloff factor. $\Delta\Omega$ in (12) is the solid angle subtended by the camera which is determined by the area of the lens S and the distance d from the centroid of the triangular facet to the camera. If shadows are present on the surface the situation is somewhat more complex. In this paper we assume that there are no shadows or occlusions present in the images.

The Area $\mathcal{A}$ of the triangle and the orientation angles in equation 12 can be calculated in terms of the vertices of the triangle, $\mathbf{P_i}$, see figure 1, as follows:

$$\begin{aligned} &\hat{\mathbf{n}} \cdot \hat{\mathbf{z}}^s = \cos\alpha^s, \quad \hat{\mathbf{n}} \cdot \hat{\mathbf{z}}^v = \cos\alpha^v, \\ &\hat{\mathbf{n}} = \frac{\mathbf{v_{i0,i1}} \times \mathbf{v_{i1,i2}}}{2\mathcal{A}}, \quad \mathbf{v_{i,j}} = \mathbf{P_j} - \mathbf{P_i} \end{aligned} \tag{13}$$

Here $\hat{\mathbf{n}}$ is a unit normal to the triangular facet and $\mathbf{v_{i,j}}$ are vectors of the edges of the triangle.

4 Computing the Derivative Matrices

To compute the MAP estimates of $\{\mathbf{z}, \rho\}$ and Θ we must compute both the image $\hat{I}(\mathbf{z}, \rho, \Theta)$ and the derivative matrices $\mathbf{D_z}$, $\mathbf{D}_\rho$ and $\mathbf{D}_\Theta$.

The derivatives with respect to the albedo values can easily be derived from equations 10 and 12. Note that because $0 < \rho < 1$, in practice, we work with transformed albedo values, where $\rho \to \log(\rho/(1-\rho))$.

Denoting by u the component of $\mathbf{z}$ or Θ that we are currently considering, the pixel intensity derivatives with respect to u have two components

$$\frac{\partial \hat{I}_p}{\partial u} = \sum_{\triangle} \left(f^p_{\triangle}\, \frac{\partial \Phi_{\triangle}}{\partial u} + \Phi_{\triangle} \frac{\partial f^p_{\triangle}}{\partial u} \right) \tag{14}$$

The first component is due to changes in angle – as the height of a vertex changes, the normal to the facet changes, and so the derivative has a component due to the change in angle between the normal and the sun direction; as the camera changes position, the angle between the normal and the ray to the camera changes.

Consider first $\partial\Phi_\triangle/\partial\Theta_i$. We neglect the derivatives with respect to the fall-off angle, θ, as their contribution will be small, and so it is clear from equation 12 that the derivative with respect to any of the camera orientation angles is zero.

The derivative with respect to the camera position parameters is given by

$$\begin{aligned}\frac{\partial\Phi_\triangle}{\partial\Theta_i} &\propto \frac{\partial}{\partial\Theta_i}\cos\alpha^v \\ &= \frac{\hat{\mathbf{n}}}{v}(\hat{z}_i - \hat{z}_v(\hat{z}_v.\hat{z}_i))\end{aligned} \tag{15}$$

where $\mathbf{v}$ is the vector from the triangle to the camera, $v = |\mathbf{v}|$, Θ_i are the three components of the camera position, $\hat{z}_i$ are unit vectors in the three coordinate directions and $\hat{z}_v = \mathbf{v}/v$ (see figure 1).

Consider now the derivative with respect to the height of one of the mesh vertices, $z_{\mathbf{i}}$. The flux derivative, $\partial\Phi/\partial z_{\mathbf{i}}$, can be computed directly from the coordinates of the triangle vertices and the camera position using equations 12 and 13. For the surface triangle with vertices $(\mathbf{P}_{\mathbf{i_0}}, \mathbf{P}_{\mathbf{i_1}}, \mathbf{P}_{\mathbf{i_2}})$ the flux derivative with respect to the z component of the vertex $\mathbf{P}_{\mathbf{i_0}}$ equals

$$\frac{\partial\Phi}{\partial z_{\mathbf{i_0}}} = \frac{1}{2}\rho(\mathbf{P}_{\mathbf{i_2}} - \mathbf{P}_{\mathbf{i_1}}) \times \hat{\mathbf{z}} \cdot \mathbf{g}\ \frac{S}{d^2}, \tag{16}$$

where

$$\mathbf{g} = \mathcal{I}_s(\hat{\mathbf{z}}_v \cos\alpha_s + \hat{\mathbf{z}}_s \cos\alpha_v - \hat{\mathbf{n}} \cos\alpha_s \cos\alpha_v) + \mathcal{I}_a\hat{\mathbf{z}}_v$$

and $\hat{\mathbf{z}}$ is a unit normal in the vertical direction.

For a triangle that projects entirely within a pixel, this completes the derivative computation – the second term in equation 14 is the derivative of the *fractional area* of the triangle that projects into the pixel.

4.1 Fractional Area Derivatives

When the height of a vertex, z, changes, its projection on the image plane, $\bar{\mathbf{P}}$, also moves, by $\delta\bar{\mathbf{P}}$. This gives rise to a change $\delta\bar{A}_\triangle$ in the area of the projection of the triangle, and also the change $\bar{A}_{\text{polygon}}$ in the polygon area. It follows from equation 11 that

$$\frac{\partial f^p_\triangle}{\partial z_{\mathbf{i_0}}} = \frac{1}{\bar{A}_\triangle}\left(\frac{\partial\bar{A}_{\text{polygon}}}{\partial\bar{\mathbf{P}}_{\mathbf{i_0}}} - f^p_\triangle\frac{\partial\bar{A}_\triangle}{\partial\bar{\mathbf{P}}_{\mathbf{i_0}}}\right)\frac{\partial\bar{\mathbf{P}}_{\mathbf{i_0}}}{\partial z_{\mathbf{i_0}}}. \tag{17}$$

where the point displacement derivative $\partial\bar{\mathbf{P}}_{\mathbf{i_0}}/\partial z_{\mathbf{i_0}}$ will be given later.

When the camera parameters change, the positions of the projections of the mesh vertices into the image plane will also move. The derivative of the fractional area is given by

$$\frac{\partial f^p_\triangle}{\partial\Theta_i} = \frac{1}{\bar{A}_\triangle}\sum_{\mathbf{j}=\mathbf{i}_0,\mathbf{i}_1,\mathbf{i}_2}\left(\frac{\partial\bar{A}_{\text{polygon}}}{\partial\bar{\mathbf{P}}_{\mathbf{j}}} - f^p_\triangle\frac{\partial\bar{A}_\triangle}{\partial\bar{\mathbf{P}}_{\mathbf{j}}}\right)\frac{\partial\bar{\mathbf{P}}_{\mathbf{j}}}{\partial\Theta_i}. \tag{18}$$

The point displacement derivatives will be detailed below.

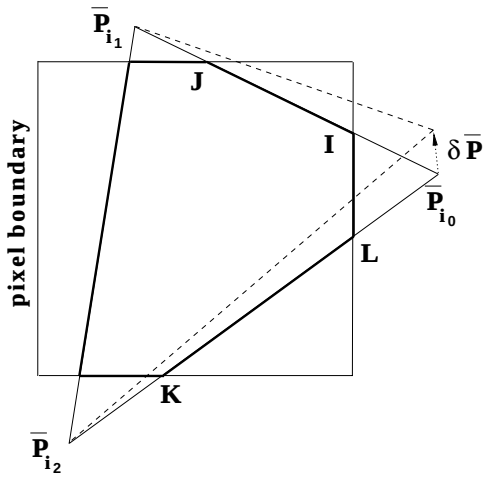

Figure 2: The intersection of the projection of a triangular surface element $(\mathbf{i_0}, \mathbf{i_1}, \mathbf{i_2})$ onto the pixel plane with the pixel boundaries. Bold lines corresponds to the edges of the polygon resulting from the intersection. Dashed lines correspond to the new positions of the triangle edges when point $\mathbf{P}_{\mathbf{i}_0}$ is displaced by $\delta\mathbf{P}$

Thus, the task of computing the derivative of the area fraction given in equation 18 is reduced to the computation of $\partial\bar{A}_{\triangle}/\partial\bar{\mathbf{P}}_{\mathbf{j}}$ and $\partial\bar{A}_{\text{polygon}}/\partial\bar{\mathbf{P}}_{\mathbf{j}}$. Note that the intersection of a triangle and a pixel for a rectangular pixel boundary can, in general, be a polygon with 3 to 7 edges with various possible forms. However the algorithm for computing the polygon area derivatives that we have developed is general, and does not depend on a particular polygon configuration. The main idea of the algorithm can be described as follows. Consider, as an example, the polygon shown in figure 2 which is a part of the projected surface triangle with indices $\mathbf{i_0}, \mathbf{i_1}, \mathbf{i_2}$. We are interested in the derivative of the polygon area with respect to the point $\bar{\mathbf{P}}_{\mathbf{i}_0}$ that connects two edges of the projected triangle, $(\mathbf{P}_{\mathbf{i}_2}, \mathbf{P}_{\mathbf{i}_0})$ and $(\mathbf{P}_{\mathbf{i}_0}, \mathbf{P}_{\mathbf{i}_1})$. These triangular edges contain segments $(\mathbf{I}, \mathbf{J})$ and $(\mathbf{K}, \mathbf{L})$ that are sides of the corresponding polygon. It can be seen from figure 2 that when the point $\bar{\mathbf{P}}_{\mathbf{i}_0}$ is displaced by $\delta\bar{\mathbf{P}}_{\mathbf{i}_0}$ the change in the polygon area is given by the sum of two terms

$$\delta\bar{A}_{\text{polygon}} = \delta A_{\mathbf{I},\mathbf{J}} + \delta A_{\mathbf{K},\mathbf{L}}$$

These terms are equal to the areas spanned by the two corresponding segments taken with appropriate signs. Therefore the polygon area derivative with respect to the triangle vertex $\bar{\mathbf{P}}_{\mathbf{i}_0}$ is represented as a sum of the two "segment area" derivatives for the two segments adjacent to a given vertex. Using straightforward geometrical arguments one can calculate the areas $\delta A_{\mathbf{I},\mathbf{J}}$ and $\delta A_{\mathbf{K},\mathbf{L}}$ to first order in the displacement $\delta\bar{\mathbf{P}}_{\mathbf{i}_0}$. Then the polygon area derivative can be written in the following form:

$$\frac{\partial\bar{A}_{\text{polygon}}}{\partial\bar{\mathbf{P}}_{\mathbf{i}_0}} = \frac{1}{2}\,\hat{\sigma}\cdot\mathbf{W}, \quad \hat{\sigma} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{19}$$

The unit antisymmetric matrix $\hat{\sigma}$ performs a -$\pi/2$ rotation in the image plane and vector $\mathbf{W}$ equals

$$\begin{aligned}\mathbf{W} &= \left[(1-R_{\mathbf{I}}^2)-R_{\mathbf{J}}^2\right]\left(\bar{\mathbf{P}}_{\mathbf{i_0}}-\bar{\mathbf{P}}_{\mathbf{i_2}}\right) \\ &\quad+\left[(1-R_{\mathbf{K}}^2)-R_{\mathbf{L}}^2\right]\left(\bar{\mathbf{P}}_{\mathbf{i_1}}-\bar{\mathbf{P}}_{\mathbf{i_0}}\right). \end{aligned} \tag{20}$$

The ratio factors R determine the positions of the intersection points $\mathbf{I},\mathbf{J},\mathbf{K},\mathbf{L}$ on the edges of the triangle (see figure 2).

$$\begin{aligned} R_{\mathbf{I}} &= \frac{\left|\mathbf{I}-\bar{\mathbf{P}}_{\mathbf{i_0}}\right|}{\left|\bar{\mathbf{P}}_{\mathbf{i_1}}-\bar{\mathbf{P}}_{\mathbf{i_0}}\right|}, \quad R_{\mathbf{J}} = \frac{\left|\mathbf{J}-\bar{\mathbf{P}}_{\mathbf{i_1}}\right|}{\left|\bar{\mathbf{P}}_{\mathbf{i_1}}-\bar{\mathbf{P}}_{\mathbf{i_0}}\right|}, \\ R_{\mathbf{K}} &= \frac{\left|\mathbf{K}-\bar{\mathbf{P}}_{\mathbf{i_2}}\right|}{\left|\bar{\mathbf{P}}_{\mathbf{i_0}}-\bar{\mathbf{P}}_{\mathbf{i_2}}\right|}, \quad R_{\mathbf{L}} = \frac{\left|\mathbf{L}-\bar{\mathbf{P}}_{\mathbf{i_0}}\right|}{\left|\bar{\mathbf{P}}_{\mathbf{i_1}}-\bar{\mathbf{P}}_{\mathbf{i_0}}\right|}. \end{aligned} \tag{21}$$

Equations 19, 21, 20 are the central result of the area fraction derivative computation. It is given for the general case of triangle-pixel intersection where two edges of triangle adjacent to the vertex $\mathbf{P}_{\mathbf{i_0}}$ each have two intersection points. Note that pairs of intersection points, $\mathbf{I},\mathbf{J}$ and $\mathbf{K},\mathbf{L}$ are defined in a unique way if one considers the triangle edges in counterclockwise order. Therefore equations 19-21 can be applied to all possible intersection cases. For example, assume that all three triangle vertices are projected inside the pixel. In this case intersection point $\mathbf{K}$ has merged with $\mathbf{P}_{\mathbf{i_2}}$, points $\mathbf{L}$ and $\mathbf{I}$ have merged with $\mathbf{P}_{\mathbf{i_0}}$ and $\mathbf{J}$ with $\mathbf{P}_{\mathbf{i_1}}$. Then in equation 21 we should put

$$R_{\mathbf{K}} = R_{\mathbf{L}} = R_{\mathbf{I}} = R_{\mathbf{J}} = 0. \tag{22}$$

In this case polygon area derivative in equation 21 is reduced to the derivative of the full area of the projected triangle

$$\frac{\partial A'_{\triangle}}{\partial \bar{\mathbf{P}}_{\mathbf{i_0}}} = \frac{1}{2}\,\hat{\sigma}\cdot\left(\mathbf{P}_{\mathbf{i_1}}-\mathbf{P}_{\mathbf{i_2}}\right). \tag{23}$$

The general rule for computing the ratio factors $R_{\mathbf{I},\mathbf{J},\mathbf{K},\mathbf{L}}$ can be formulated as follows:

- If point $\mathbf{P}_{\mathbf{i_0}}$ lies inside of the pixel one should set in equation 21 ratio factors $R_{\mathbf{L}}=0$ and $R_{\mathbf{I}}=0$.
- If point $\mathbf{P}_{\mathbf{i_2}}$ lies inside of the pixel then one sets $R_{\mathbf{K}}=0$.
- If $\mathbf{P}_{\mathbf{i_1}}$ lies inside then $R_{\mathbf{J}}=0$.

This describes all possible intersection cases and provides a full description for the area fraction derivative (18).

We now consider the point displacement derivatives.

4.2 Derivatives of the position of the projection of a point on the image plane.

The pinhole camera model [4] gives

$$\hat{u}_l = \frac{[\mathbf{AR}(\mathbf{P}-\mathbf{t})]_l}{[\mathbf{R}(\mathbf{P}-\mathbf{t})]_3} \tag{24}$$

where **R** is the rotation matrix from world to camera coordinates, **t** is the translation of between camera and world coordinates and **A** is the matrix of camera internal parameters [5], $\hat{u}$ is the projection of point **P** onto the image plane and $l = \{1, 2\}$ indexes the x and y components. In the numerical experiments presented here we assume that the internal camera parameters are known, and further that the image plane axes are perpendicular, and that the principle point is at the origin. This reduces **A** to a diagonal matrix with elements $(k_1, k_2, 1)$, where $k_1 = -\mathrm{f}/l_x$, $k_2 = -\mathrm{f}/l_y$. Where f is the focal length of the lens and l_x and l_y are the dimensions of the pixels in the retinal plane.

The rotation matrix **R** can be written in terms of the *Rodrigues* vector [4] $\boldsymbol{\rho} = (\rho_1, \rho_2, \rho_3)$ which defines the axis of rotation, and $\theta = |\rho|$ is the magnitude of the rotation. (Clearly ρ can be written in terms of the camera position, the look-at point and the view-up vector.)

$$\mathbf{R} = \mathbf{I} - \mathcal{H}\frac{\sin\theta}{\theta} + \mathcal{H}^2\frac{(1-\cos\theta)}{\theta^2} \tag{25}$$

where

$$\mathcal{H} = \begin{pmatrix} 0 & -\rho_3 & \rho_2 \\ \rho_3 & 0 & -\rho_1 \\ -\rho_2 & \rho_1 & 0 \end{pmatrix}. \tag{26}$$

Let $H = \mathcal{H}/\theta$ and $r_i = r_i/\theta$ then

$$\begin{aligned} \frac{\partial \mathbf{R}}{\partial \rho_i} &= -\mathcal{H}_i\frac{\sin\theta}{\theta} + (H\mathcal{H}_i + \mathcal{H}_i H)\frac{(1-\cos\theta)}{\theta} \\ &\quad - H\left(\cos\theta - \frac{\sin\theta}{\theta}\right) r_i \\ &\quad + H^2\left(\sin\theta - 2\frac{1-\cos\theta}{\theta}\right) r_i \end{aligned} \tag{27}$$

where $\mathcal{H}_i = \partial\mathcal{H}/\partial\rho_i$. Then

$$\frac{\partial \hat{u}_l}{\partial \rho_i} = \left(\frac{\left[\mathbf{A}\frac{\partial \mathbf{R}}{\partial \rho_i}(\mathbf{P}-\mathbf{t})\right]_l}{[\mathbf{R}(\mathbf{P}-\mathbf{t})]_3} - \frac{[\mathbf{AR}(\mathbf{P}-\mathbf{t})]_l \left[\frac{\partial \mathbf{R}}{\partial \rho_i}(\mathbf{P}-\mathbf{t})\right]_3}{([\mathbf{R}(\mathbf{P}-\mathbf{t})]_3)^2} \right) \tag{28}$$

The derivatives with respect to the position parameters are

$$\frac{\partial \hat{u}_l}{\partial \mathbf{t}_j} = \frac{[\mathbf{AR}(\mathbf{P}-\mathbf{t})]_l [\mathbf{R}]_{3,j}}{([\mathbf{R}(\mathbf{P}-\mathbf{t})]_3)^2} - \frac{[\mathbf{AR}]_{l,j}}{[\mathbf{R}(\mathbf{P}-\mathbf{t})]_3} \tag{29}$$

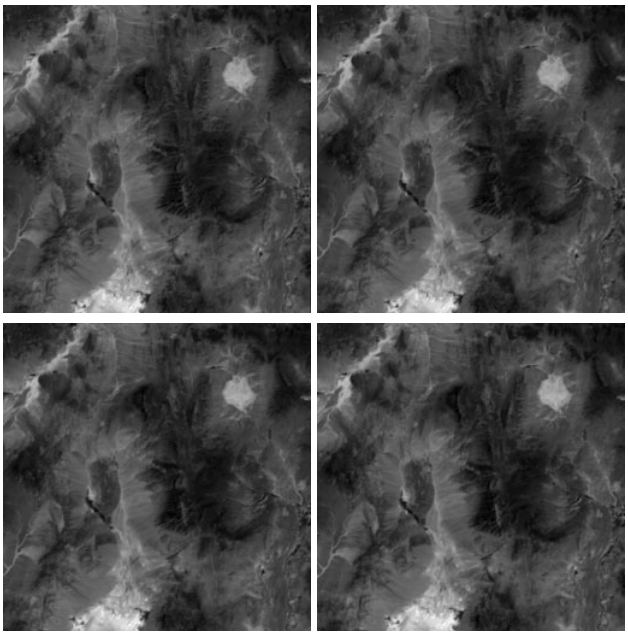

Figure 3: Four synthetic images of Duckwater, Nevada

Note that whilst in equation 28 we have given the differentials with respect to the Rodriguez vector components, when performing the minimization in equation 4 we do so in terms of the look-at point, as this is in the natural length scale of the problem and leads to better convergence. Converting the derivatives is an application of the chain rule and is not detailed here.

The derivative with respect to the heights is the same as the derivative with respect to the z-component of the camera position, but with the sign reversed, that is

$$\frac{\partial \hat{u}_l}{\partial \mathbf{z}_i} = -\frac{[\mathbf{AR}(\mathbf{P}-\mathbf{t})]_3 [\mathbf{R}]_{3,3}}{([\mathbf{R}(\mathbf{P}-\mathbf{t})]_3)^2} + \frac{[\mathbf{AR}]_{l,3}}{[\mathbf{R}(\mathbf{P}-\mathbf{t})]_3}$$

5 Results

Figure 3 shows four synthetic images of a region of Duckwater, Nevada. They were generated by rendering a synthetic surface. The surface was constructed by using the USGS Digital Elevation Model for the heights, and using the scaled intensities of a LANDSAT-TM image as surrogate albedos. The size of the surface is 301×301 points. The distance between grid points was taken to be one unit, and the heights scaled appropriately. Figure 4 shows a perspective view of the surface. The albedos have been raised to the power 0.4 to stretch the contrast and the vertical scale has been expanded. Table 1 gives the camera parameters that were used to generate the images.

An initial estimate of the camera parameters was made by using point matching [7]. We have found that the Harris corner detector [6] typically used to select features

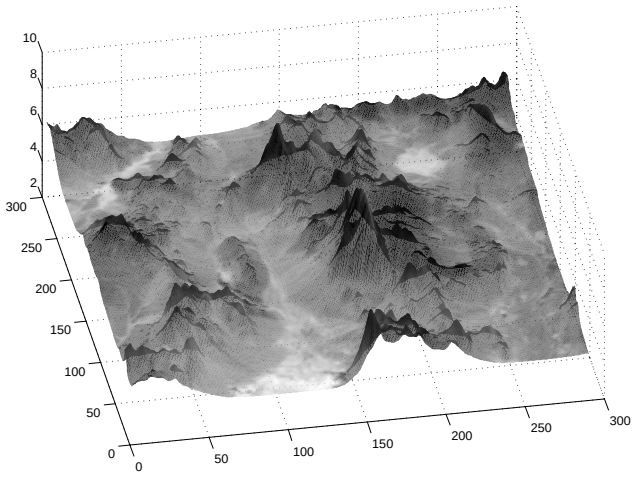

Figure 4: True surface

does not find many reliable features in he types of natural scenery we are concerned with here. Table 2 gives the parameters estimated by matching five points across the four images.

Using these estimated camera parameters, a dense surface estimate can be made. For space reasons we do not show the surface estimate, instead, in figure 6 we show the error surface. The main points to note are that

1. inaccuracies in the camera parameter estimation have resulted in an erroneous slope in the surface estimate.

2. the overall height of the surface is shifted upwards; but note that the overall shift is a small percentage (less than 0.5%) of the distance from the surface to the cameras. The overall height is only weakly determined.

3. the albedo estimates are in general quite good (the RMSE for the albedo estimate is 0.022).

Using the gradient-based, whole image, approach to camera calibration to a surface, that we have described above, we then registered the images to the surface estimate. Using the new camera parameters, we re-estimated the surface. This was iterated three times. Table 3 gives the final camera parameters, and figure 5 shows the final surface estimate. Figure 7 is the error surface and figure 8 is a section through the error surfaces. We note the following:

1. the main improvement in the camera parameter estimation is in the orientation angle, defined by the view-up vector

2. the erroneous slope has been corrected

3. the error in the global height remains

4. the estimate shows most inaccuracies close to rapid changes in albedo, for example the white (salt lake) area to the top right of the surface, where albedo and slope effects have not been completely decoupled.

6 Conclusions

In this paper we have described a system that takes a set of images and uses them to infer both the *camera parameters* and a *dense surface model*. It does this by iterative linearization of a model of the image formation process, and minimization of the error between the whole of the observed and rendered images with respect to the camera and surface parameters. We have demonstrated the convergence of this system on a set of images rendered from a model of a region of Nevada.

The system we have described has many advantages. The scale of the surface model that is estimated is decoupled from the pixel scale of the images via the rendering process. This means that the surface model scale can be chosen by the user, either on the basis of the use to which the surface model will be put, or a scale may be chosen which is best justified by the image data. This is important – if we have many low resolution images of a region, the scale of the surface model may be super-resolved (where a triangular surface element projects onto an area smaller than a pixel on the image plane). If the coverage of the surface by the images is non-uniform, we can specify a spatially-varying mesh for the surface, denser in regions where we have more images.

The information about the surface captured by the system is not just the MAP surface estimate, but also the accuracy of the estimate, represented by the inverse covariance matrix (**A** in equation 5). Knowing the inverse covariance matrix allows for recursive updates – as new images become available the information they contain can be integrated into the model. In Bayesian terminology, the posterior distribution from one set of images (defined by the MAP estimate and the inverse covariance matrix) becomes the prior for estimation with new images.

Finally, we are not restricted to only image data. If data from other sensing modalities is available (for example, laser altimetry data) then we can add a term to the likelihood (equation 2) for this data, take derivatives of a model of how this new sensor makes measurements with respect to the surface model parameters, and our surface model estimate will seamlessly integrate the multi-modal information.

References

[1] J. Foley, A. van Dam, S. Finer, and J. Hughes. *Computer Graphics, principles and practice*. Addison-Wesley, 2nd ed. edition, 1990.

[2] A.L Yuille, D. Snow, R. Epstein and P.N. Belhumeur Determining Generative Models of Objects Under Varying Illumination: Shape and Albedo from Multiple Images Using SVD and Integrability. International Journal of Computer Vision, 35(3), 203-222, 1999.

[3] J. Bernardo and A. Smith. *Bayesian Theory*. Wiley, Chichester, New York, 1994.

image 1	camera	$(-100, 150, 1700)$
	look at	$(150, 150, 0)$
	view up	$(0, 1, 0)$
image 2	camera	$(300, 150, 1700)$
	look at	$(150, 150, 0)$
	view up	$(0, 1, 0)$
image 3	camera	$(150, -100, 1700)$
	look at	$(150, 150, 0)$
	view up	$(0, 1, 0)$
image 4	camera	$(150, 345, 1700)$
	look at	$(150, 150, 0)$
	view up	$(0, 1, 0)$

Table 1: Camera parameters used to generate the images in figure 3.

image 1	camera	$(-100, 150, 1700)$
	look at	$(150, 150, 0)$
	view up	$(0, 1, 0)$
image 2	camera	$(301, 150, 1700)$
	look at	$(152, 150, 0)$
	view up	$(0.0031, 1, -0.00031)$
image 3	camera	$(151, -101, 1700)$
	look at	$(151, 149, 0)$
	view up	$(-0.0032, 0.989, 0.146)$
image 4	camera	$(153, 348, 1700)$
	look at	$(151, 151, 0)$
	view up	$(0.0021, 0.933, -0.116)$

Table 2: Camera parameters estimated using point matching. The first image was taken as a known reference.

image 1	camera	$(-100.0, 150.0, 1700.0)$
	look at	$(150.0, 150.0, 0)$
	view up	$(-0.00084, 1, 9.0 \times 10^{-5})$
image 2	camera	$(301.0, 150.0, 1700.0)$
	look at	$(151.9, 150.1, 0)$
	view up	$(0.00066, 1, -2.38 \times 10^{-5})$
image 3	camera	$(151.0, -101.0, 1700.0)$
	look at	$(151.1, 149.0, 0)$
	view up	$(0.00039, 0.989, 0.146)$
image 4	camera	$(153.0, 348.0, 1700.0)$
	look at	$(151.0, 151.0, 0)$
	view up	$(0.00055, 0.9934, -0.115)$

Table 3: Final camera parameters estimated using gradient based image error estimation on the estimated surface (3rd iteration).

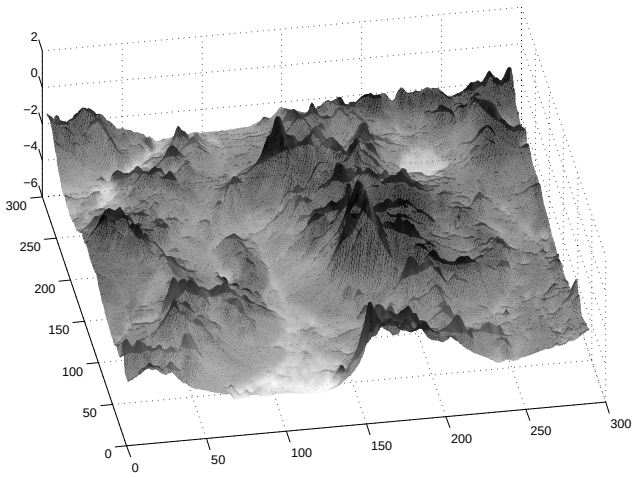

Figure 5: Inferred surface

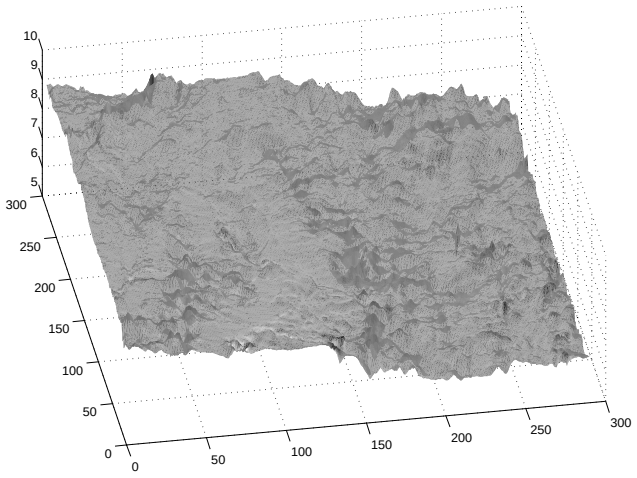

Figure 6: Error surface for the surface estimate using camera parameters from point matching

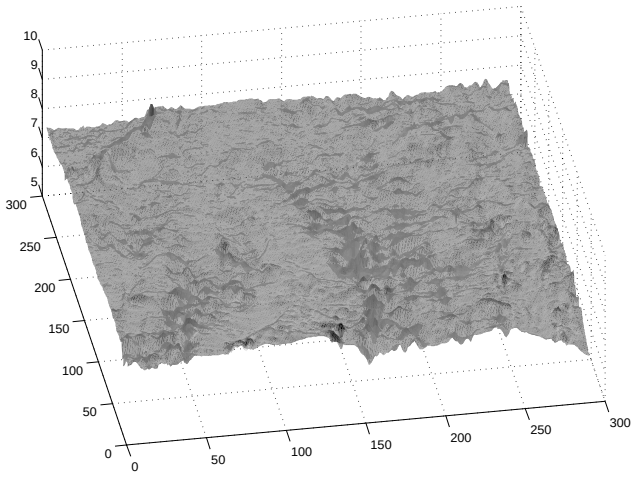

Figure 7: Error surface for the surface estimate using iteratively refined camera parameters

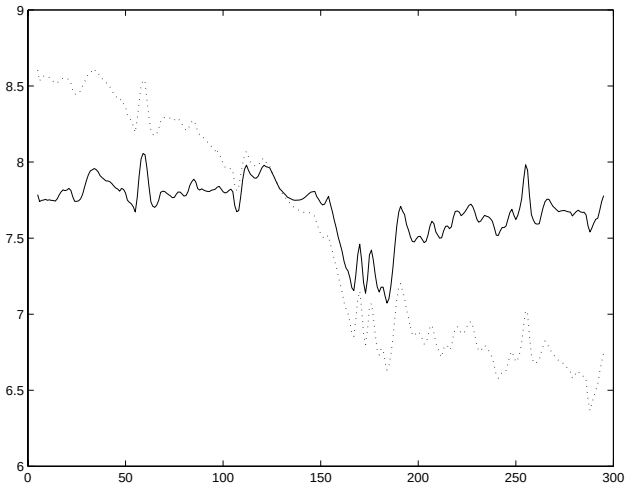

Figure 8: Section through the error surfaces. Dotted line - error of pointmatching estimate; solid line - error of the final estimate

[4] O. Faugeras. *Three-Dimensional Computer Vision*. MIT Press, 1993.

[5] Z. Zhang. A Flexible New Technique for Camera Calibration. Technical Report MSR-TR-98-71, Microsoft Research, Redmond, Washington.

[6] C. Harris. A Combined Corner and Edge Detector. Proceedings of the Alvey Vision Conference, pp 189-192, 1987.

[7] Z. Zhang, R. Deriche, O. Faugeras,and Q.T. Luong. A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry. AI Journal, vol. 78, pp 87-119, 1994.